Microsoft **Engineering Excellence**

Journey of Enlightenment: The Evolution of Development at Microsoft

Eric Brechner, Ph.D.

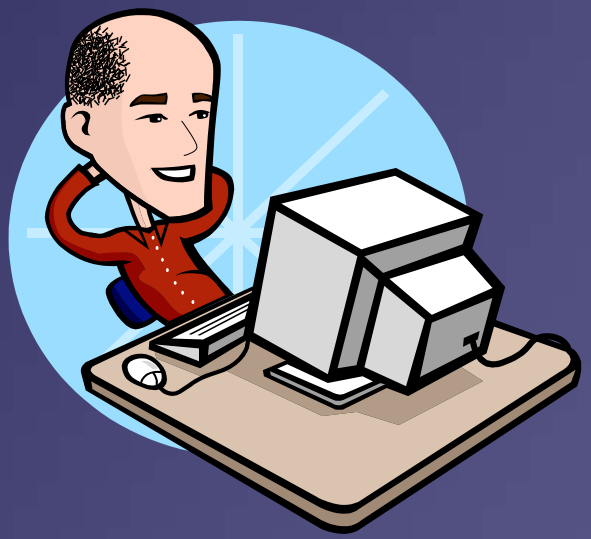
Director, Development Excellence

Microsoft Corporation

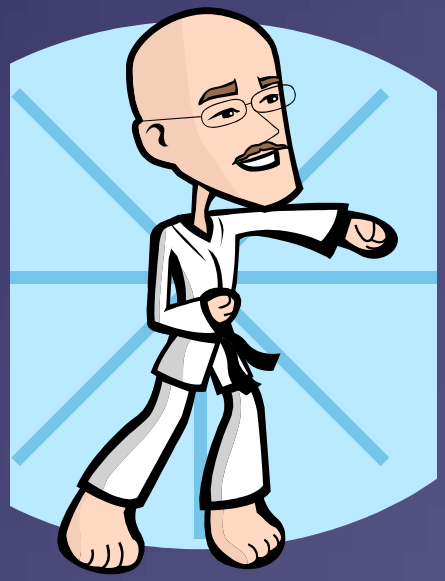
Microsoft

Software is easy

- I'll bet half the people in this room know exactly how they would solve "Software Everywhere"
- Aside from that abstract belief, their ideas probably have little in common



People are hard



- Why?
  - People see problems in different ways
  - People have different goals
  - People learn and change their minds
- Those three issues are at the heart of Software Everywhere, as well as software development

Customers are really hard

- Customers are people
  - Different viewpoints, different goals, and fickle needs
- Their expectations change over time
  - Yesterday: only techies used software
  - Today: everyone uses software all the time
  - Yesterday: the Internet was like a 60s commune
  - Today: the Internet is like, well, the real world
  - Yesterday: “Give me features fast, I’ll make it work”
  - Today: “If it doesn’t work, neither do I”

Software development at Microsoft

- The emergence of .NET
  - Interconnectivity
  - Ubiquity
- Agility with discipline
  - Old school Microsoft
  - Agile influence
  - Just enough process
  - Just enough documentation
  - More than a handshake

The emergence of .NET

- The goal: ubiquitous interconnected smart devices
- Requires device-neutral programming and UI
- The tough part: ubiquitous and interconnected

Interconnectivity

- Data manipulation
- Communications
- Security
- Reliability

Ubiquity

- Ubiquity → Adoption
- Adoption by users: Common UI across devices
- Adoption by developers:
  - Support for all common languages
  - Easy development
  - Easy debugging

Agility with discipline

- 7000 developers + easy development = chaos
- Chaos != quality or customer satisfaction
- We needed to add some controls without causing a riot

Old school Microsoft

- Developers used to do everything
- Big, complex projects formalized PMs and testers
- Product unit = 30 to 60 PMs, devs, and testers
- Every product unit was independent and autonomous
- Modified waterfall model with daily integrated builds
- Widely varying use of unit tests, code reviews, and design inspections
- Microsoft today = 400+ product units
  - Windows = 100+ product units

Agile influence

- Developers love the agile concepts
  - Focus on the customer and delivering value instead of documentation, plans, and wrangling
  - Rapid development with constant customer feedback
- The platform catch: it's tough to get 100 million customers in a room once a week

Just enough process

- 100+ teams around the world working on an integrated platform simultaneously means standards
  - Standard build
  - Standard setup
  - Standard quality gates
  - Going too far means disaster
- Better coding practices (TDD, Scrum, Lean)
- As much isolation for components as possible

Just enough documentation

- The interfaces must be well designed, documented, and versioned
- Versioning of interfaces is tricky
- Design is a voyage, not a destination

More than a handshake

- You need it in writing: contracts
- Expectations and requirements for both sides
- Ownership and responsibilities for both sides
- Fallback positions for both sides
- For Web services: WSDL
- For people: e-mail or Word docs
- In both cases, trust but verify

Conclusions

- Managing a large dev team is a little like managing distributed devices
  - You need standards
  - You need contracts
  - You need change control
- Too little leads to chaos
- Too much stifles creativity and autonomy → innovation
- Be practical; use just enough; use what makes sense
- Be patient

Microsoft **Engineering Excellence**

Questions?